7-1-2006

Realistic Large-Scale Network Optimisation

Vic Grout
Glyndwr University, v.grout@glyndwr.ac.uk

Rich Picking
Glyndwr University, Wrexham, r.picking@glyndwr.ac.uk

Realistic Large-Scale Network Optimisation

Abstract
This paper considers communication network design problems that arise in the real world, with large numbers of nodes - and link and switch costs dependent upon their traffic capacity. Such costs, in turn, depend upon network topology so are not fixed at the start of, or through, any optimisation process. Realistic topological restrictions are also discussed. The limitations of conventional approaches – both constructive and search based – are noted and the requirements of practical optimisation methods explored. Two workable approaches to network design - one an established local search variant, another a more novel geometric approach – are introduced and combined. Various simple and compound algorithms, ranging from exhaustive search to fast heuristic are compared with experimental results given in conclusion.

Keywords
Large-scale network optimisation, Algorithms and heuristics

Disciplines
Computer and Systems Architecture | Digital Communications and Networking | Hardware Systems | Systems and Communications

Comments
Runner-up, 'Best paper at Conference' award, Internet Research/Emerald. This paper was presented at 6th International Network Conference (INC 2006)] 11-14 July 2006, which was held by University of Plymouth and details of the conference are available at http://www.cscan.org

Realistic Large-Scale Network Optimisation

Vic Grout, Rich Picking, Stuart Cunningham and Rich Hebblewhite,

Centre for Applied Internet Research (CAIR), University of Wales, NEWI, Wrexham, UK
{v.grout|r.picking|s.cunningham|r.hebblewhite}@newi.ac.uk

Abstract

This paper considers communication network design problems that arise in the real world, with large numbers of nodes - and link and switch costs dependent upon their traffic capacity. Such costs, in turn, depend upon network topology so are not fixed at the start of, or through, any optimisation process. Realistic topological restrictions are also discussed. The limitations of conventional approaches – both constructive and search based – are noted and the requirements of practical optimisation methods explored. Two workable approaches to network design - one an established local search variant, another a more novel geometric approach - are introduced and combined. Various simple and compound algorithms, ranging from exhaustive search to fast heuristic are compared with experimental results given in conclusion.

Keywords

Large-scale network optimisation, Algorithms and heuristics

1. Introduction: Conventional Network Optimisation

It is a common mistake to consider the topological network design problem (TNDP) for fixed networks in general, and communication networks in particular, well-solved. In the standard formulation, n nodes are to be interconnected with c_{ij} representing the cost of connecting node i directly to node j. The problem is then to find a connecting set of links minimising

$$C^* = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} \,. \tag{1}$$

Early constructive algorithms (e.g. Prim, 1957) solve the problem to optimality in its unconstrained form and produce *Minimal Spanning Tree* (MST) solutions (Figure 1.(a)). Capacity constraints can be applied although the problem then becomes NP-hard (Garey and Johnson, 1979) and the necessarily adapted heuristics (e.g. Kershenbaum and Chou, 1974) only yield approximate solutions. However, with an initial solution in place, various classes of local search heuristics such as tabu-search (Glover and Laguna, 1993), simulated annealing (Aarts and Lenstra, 1997), genetic algorithms (Winter *et al.*, 1995) or Ant Colony Algorithms (Dorigo *et al.*, 1999) can be applied to perturbate parts of the solution to look for improvement. A common, but unrealistic, approach is to formulate the problem in Linear or Integer Programming (Verkhovsky and Polyakov, 2003). This paper begins by outlining the practical shortcomings of the TNDP formulation and its associated algorithmic solutions. It then discusses the requirements of a real-world fixed network design optimisation process and introduces and compares various *effective* solutions.

2. Limitations of Conventional Optimisation

There are two objections to the MST solution to the TNDP. Firstly, the connecting network will have long, inefficient paths, even between geographically close nodes and, secondly, the solution network will be extremely vulnerable to component failure. There is no redundancy

- a fault at any node or link disconnects the network. However, there are also difficulties associated with this simplistic notion of cost, independent of the method of solution. Only link (e.g. transmission) costs are considered; node (e.g. switching) costs are ignored. Also, traffic will flow, possibly asymmetrically, in both directions on a link - the structure of link costs should reflect this. Finally, costs are taken as fixed in the statement of the problem and throughout any optimisation process, irrespective of network topology. This final point warrants further explanation. The true cost of a link will depend partly on its capacity - the level of traffic it can handle. A similar variance applies to switches. The cost of a link or switch consequently depends on the solution topology so cannot be fixed for the duration of the optimisation process.

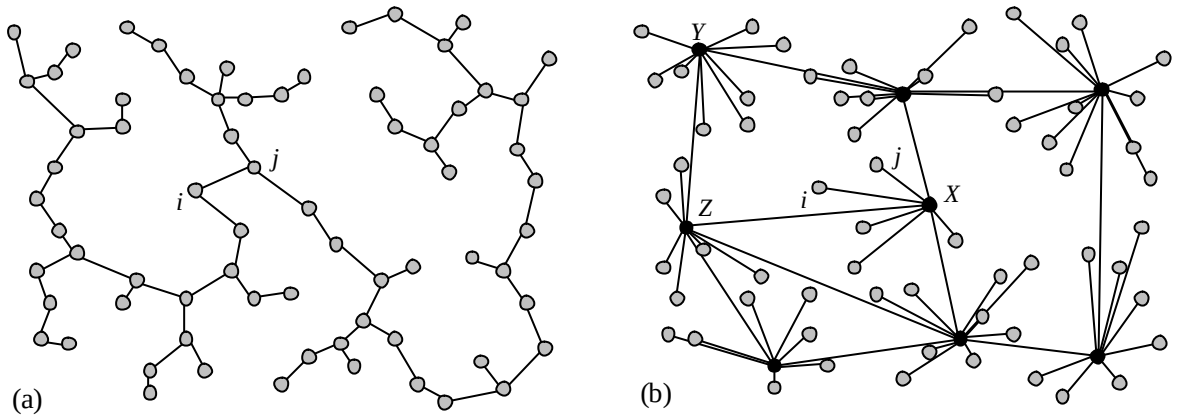

Figure 1. MST and practical network design.

For large numbers of nodes a more realistic network topology is shown in Figure. 1.(b). A subset of nodes (switches) is chosen to concentrate and relay traffic among the remainder through a mesh or partial-mesh *core* network. The maximum path length between any node pair is reduced significantly and there is some tolerance of failure, at least in the core network, provided by redundant links. We take this as our model for a practical network design in this paper. Other forms are possible of course, such as constrained full-mesh or star core networks, multiply-connected non-switch nodes or multi-level networks. The techniques discussed in this paper extend without difficulty to these variants.

3. A Practical Formulation for Network Cost

Link costs remain variable, however, and switch costs should be considered. If we adopt the convention of using uppercase characters for switches and lowercase for non-switches then, in general, $c_s(i)=0$, $c_s(X)>0$, $c_l(i,j)=0$ and, where the link in question is present, $c_l(X,Y)>c_l(i,X)>0$, where c_s and c_l are the costs of switches and links respectively. More precisely, if a link L carries traffic t over a distance d then $c_l(L) = f_l(t,d)$. If a switch S processes traffic T then $c_s(S) = f_s(T)$. f_l and f_s may be any well-defined functions, dependent upon the underlying technology. Define t_{ij} to be the traffic between end-points i and j, that is the traffic originating at i and destined for j. Define d_{ij} to be the 'distance' between i and j. This may the Euclidean straight line ($d_{ij} = [(x_i-x_j)^2+(y_i-y_j)^2]^{½}$ where (x_i,y_i) and (x_j,y_j) are the Cartesian coordinates of i and j) or weighted to reflect local factors. If a link is infeasible then $d_{ij}=¥$. The cost of a link from a non-switch i to its *parent* switch X is then given by

$$c_l(i,X) = f_l(\sum_{j=1}^{n} t_{ij}, d_{iX}) \qquad (2)$$

with a corresponding cost $c(X, i)$ in the other direction. Define G_X to be the set of nodes with

X as their parent in a given configuration/solution. Then the cost of the switch *X* is given by

$$c_s(X) = f_s(\sum_{i\in\Gamma_X}\sum_{j=1}^{n}(t_{ij}+t_{ji}))\,. \tag{3}$$

For a fully-connected core network, i.e. with a link between each switch pair *(X, Y)*, the cost of the link *(X, Y)* is given by

$$c_l(X,Y) = f_l(\sum_{i\in\Gamma_X}\sum_{j\in\Gamma_Y} t_{ij}, d_{XY}) \tag{4}$$

with an equivalent cost $c_l(Y,X)$ in reverse. Define $W_{XY} = 1$ if there is a link between *X* and *Y*. $W_{XY} = 0$ otherwise. (The node sets G_X, G_Y, …, and the connection matrix $W = (W_{XY})$ fully describe any given solution.) The total cost of the (fully-connected) network can then be calculated as

$$c^* = \sum_X \left[\begin{array}{l} \sum_{i\in\Gamma_X}(c_l(i,X)+c_l(X,i)) \\ +\,c_s(X) \\ +\sum_Y c_l(X,Y) \end{array} \right]. \tag{5}$$

If the link from switch *X* to switch *Y* is not present ($W_{XY} = 0$) this results in a saving of

$$f_l(\sum_{i\in\Gamma_X}\sum_{j\in\Gamma_Y} t_{ij}, d_{XY})\,. \tag{6}$$

However its traffic must be redirected via switches Z_1, Z_2, … The cost of each affected switch, Z_1, Z_2, … will increase to

$$\begin{array}{l} f_s(\sum_{i\in\Gamma_{Z_1}}\sum_{j=1}^{n}(t_{ij}+t_{ji}) + \sum_{i\in\Gamma_X}\sum_{j\in\Gamma_Y} t_{ij}), \\ f_s(\sum_{i\in\Gamma_{Z_2}}\sum_{j=1}^{n}(t_{ij}+t_{ji}) + \sum_{i\in\Gamma_X}\sum_{j\in\Gamma_Y} t_{ij}),\ldots \end{array} \tag{7}$$

and link costs to

$$\begin{array}{l} f_l((\sum_{i\in X}\sum_{i\in Z_1} t_{ij} + \sum_{i\in X}\sum_{i\in Y} t_{ij}), d_{XZ_1}), \\ f_l((\sum_{i\in Z_1}\sum_{i\in Z_2} t_{ij} + \sum_{i\in X}\sum_{i\in Y} t_{ij}), d_{Z_1Z_2}), \\ \ldots \\ f_l((\sum_{i\in Z_r}\sum_{i\in Y} t_{ij} + \sum_{i\in X}\sum_{i\in Y} t_{ij}), d_{Z_rY}) \end{array} \tag{8}$$

for each capacity-enlarged link, *(X, Z_1), (Z_1, Z_2), …, (Z_r, Y)* where *r* is the degree of redirection for *(X, Y)* ($r=0$ Û $W_{XY}=1$). The calculation is repeated for each *(X, Y)* with $W_{XY} = 0$. (If $W_{XY} = 0$ implies $W_{YX} = 0$ then the adjustments in (6, 7 & 8) are replicated in reverse but this is not assumed here.) The total network cost c^* can be recalculated accordingly. The removal of a link will result in an overall saving if appropriate spare capacity can be found on the switches and links through which its traffic is redirected. (A distinction is made here between *redirection* and *rerouting*. Redirection is part of the topological design process by which required link capacities are estimated. Rerouting is a dynamic process taking place in real time on network switches. The use of redirection in design does not prohibit dynamic

rerouting in operation.) c^*, however, is a complex calculation, based on link costs that vary with network topology. Significantly, small changes to a topology (such as moving a node to a different parent switch) have consequential effects across the network and require a full re-evaluation of the total cost. Conventional local search techniques work well when the effects of a local change can be calculated locally in terms of a change in cost (such as the insertion/removal of a link of fixed cost). Their complexity is increased if the cost function must be recalculated for each perturbation and their power diminishes rapidly

4. Realistic Local Search

Theoretical search routines do not work well for the variable cost problem outlined here. There are n^{n-2} possible trees on n nodes (Moon, 1970) and a number of connected networks given recursively by

$$\Phi_n = 2^{n(n-1)/2} - \sum_{m=1}^{n-1} \frac{(n-1)!\Phi_m 2^{(n-m)(n-m-1)/2}}{(m-1)!(n-m)!} \tag{9}$$

(Grout, 2005a). Both expressions are exponential, implying that exhaustive search is impractical for larger n. An approach favoured by practical network designers, although its origins are uncertain, is outlined in the following algorithm.

```
DD(n):
Make every node a switch                                    // Initial solution
repeat
        Connect switches as a full-mesh
        Ds = 0
        Calculate c*                                        // Equation 5
        for each switch X do                                // Look to drop switches
            begin
                Calculate c*(X)                             // Equation 5
                repeat
                        Dl = 0
                        for each link (Y,Z) do              // Look to drop links
                            begin
                                Calculate D = c*(X)-c*(X,Y,Z)   // Equations 6,7,8
                                if D > Dl then
                                        Y* = Y, Z* = Z, Dl = D
                            end
                        if Dl > 0 then
                            begin
                                Remove link (Y*,Z*)         // Drop 'worst' link
                                c*(X) = c*(X,Y*,Z*)
                            end
                until
                        Dl = 0                              // No further link savings
                D = c* - c*(X)
                if D > Ds then
                        X* = X, Ds = D
            end
        if Ds > 0 then
                Remove switch X*                            // Drop 'worst' switch
until
        Ds = 0                                              // No further switch savings
```

Starting from a full-switch/fully-connected network, the 'Double-Drop' (DD) algorithm tries candidate switches for removal from the current solution. With each trial switch removed, links are experimentally dropped in a similar manner. The algorithm is essentially 'greedy' but in a nested, local-search form. The network cost c^* and perturbated costs $c^*(X)$, $c^*(Y,Z)$ and $c^*(X,Y,Z)$ (the cost of the 'current' network without the switch X and/or the link (Y,Z)) are calculated as in Section 3. There is an assumption that nodes are connected to their nearest switch. DD is a practical algorithm in that it deals with costs that vary with network topology. Its simple structure also minimises search iterations. Its major drawbacks are that:

it is unlikely to be particularly accurate since it removes switches and links in an entirely greedy manner with no consideration for a wider search neighbourhood, it is still computationally complex in its consideration of all combinations of node and link drops at each stage and its complexity is increased further by the need to completely recalculate the cost function for each perturbation. A natural extension to the DD process, to overcome the shortcomings of greedy search, is to introduce larger search neighbourhoods through (e.g.) tabu search and simulated annealing. However, such refinements, whilst addressing the first problem, simply compound the second. For moderately-sized problems (n), unrefined DD has typically proved be the only viable search process.

5. Representative Reduction

An alternative design method is proposed for large networks that eliminates a large number of iterations, branches and cost calculations. It uses the traffic values t_{ij} and distances d_{ij} to geometrically reduce the network in size. 'Conventional' optimisation then proceeds on the reduced representative version. Define the *weight* of each node to be its total traffic load:

$$w_i = \sum_{j=1}^{n} (t_{ij} + t_{ji}) \tag{10}$$

and note that this value is constant for any solution topology. We also require each node i to be defined by its Cartesian coordinates, (x_i,y_i). Then define a single *reduction step*, *RS(m)*, acting on m nodes, as:

```
RS(m):
min = MaxVal                                          // Some arbitrarily large value
for each node pair i, j (1£i¹j£m) do
        if dij < min then                             // Find closest pair
                i* = i, j* = j, min = dij
xk = (wi*xi* + wj*xj*) / (wi* + wj*)
yk = (wi*yi* + wj*yj*) / (wi* + wj*)
wk = wi* + wj*                                        // Replace by a single node
for each node, ? (? ¹ i*,j*) do                       // with representative traffic,
    begin                                             // coordinate and distance
        dk? = (wi*di*? + wj*dj*?) / (wi* + wj*)       // characteristics
        d?k = (wi*d?i* + wj*d?j*) / (wi* + wj*)
    end
```

RS(m) finds the closest two nodes, as defined by distances d_{ij} and replaces them by a single, representative node, biased by the weights w_i and w_j. The original m nodes are replaced by a representative *m-1* in this single step. *RS(m)* is the essential component in a compound algorithm that can perform conventional optimisation on a network problem of reduced size. If *RS(m)* is repeated $n - q$ times, the original network problem of size n will be replaced by a representative one of size q. These q nodes can be used in three ways to approximate an optimum solution – described in the next section. The complexity of the reduction process, a sequence of matrix searches, is bounded above by $O(n^3)$.

6. Practical Optimisation following Reduction

Assuming the intention is to site switches at existing locations, define the step *Rel(q)* to be the process of relocating the q representative nodes to their nearest true node. If greenfield sites are permitted then the step may be omitted from the final process. For any given computer upon which optimisation is to be performed (i.e. its processor power) we define *optimisation limit values*. n_{ES} is the maximum number of nodes for which exhaustive search is feasible and n_{DD} the maximum number of nodes for which double-drop is feasible. Clearly, $n_{DD} > n_{ES}$ but actual values depend on the time available. For a given (time) limit, the value of n_{ES} may be derived empirically or calculated exactly from the known complexity of the

exhaustive search process. The double drop algorithm, however, is indefinitely iterative so n_{DD} is best derived by experimentation. Three compound heuristics are outlined:

6.1. Reduction to Exhaustive Search (RES)

This is a simple, intuitive process. Reduce the number of nodes to n_{ES}, relocate to true positions and optimise to find switches and the core network through exhaustive search.

6.2. Reduction to Double-Drop (RDD)

This is equally simple. Reduce the number of nodes to n_{DD}, relocate to true positions and perform double-drop (DD) optimisation to find switches and the core network.

6.3. Reduction to Switch Location (RSL)

This is not so straightforward. The idea is as follows. Reduce the number of nodes by one each time, immediately relocate to true positions (a single step only for the new node), explicitly make each node a switch and optimize on the core network only. Calculate cost (Equations 5, 6, 7 & 8). Repeat while cost decreases. However, this would be an extremely complex approach. Evaluating each of the Φ_m core networks for each decreasing value of m switches (starting with $m = n$) is comparable with exhaustive search for complexity. To avoid this, we adopt the heuristic approach of only evaluating the cost of a fully-connected (mesh) core network. Each algorithm is as follows:

```
RES:
m = n
repeat
    RS(m)
until
    m = n_ES
Rel(m)
ES(m)
```

```
RDD:
m = n
repeat
    RS(m)
until
    m = n_DD
Rel(m)
DD (m)
```

```
RSL:
m = n
repeat
    RS(m)
    Rel(m)
    COpt(m)
until
    co(m) > co(m+1)
m = m+1
Opt(m)
```

COpt(m) is the process of finding the m switches with the cheapest full-mesh core network and *co(m)* is the cost of this core network. The *optimal* core network is only calculated (by exhaustive search) for the final switch set (*Opt(m)*).

7. Testing and Results

The algorithms introduced in this paper are compared here. Two types of test instances were used: computer-generated and real-world. It is known (Grout, 2005b) that certain algorithms can favour problem instances with parameters taken from particular statistical distributions so every attempt has been made to consider a variety of situations and characteristics.

7.1. Computer-Generated Instances (CG)

Random generation of test instances is straightforward but must be appropriate and realistic. Just over *4,000* instances were produced with numbers of nodes (n) between *10* and *100,000*. Node positions were randomly taken from the *[0,1]* unit square but with reference to between *0* and *25 cluster points* (*cp*) and a *cluster coefficient* (*cc*) of between *0* and *1*. A *cc* of *1* forces all nodes to be coincident on cluster points. A *cc* of *0* allows nodes to be placed anywhere – a uniform distribution across the unit square. *cp* and *cc* were randomized uniformly. End-to-end traffic figures between each node pair were independently randomized on the interval *[0,1]* according to (both, separately) a uniform distribution (U) and a normal distribution (N_s) with mean *0.5* and standard deviation (s) between *0.005* and *0.25*. Realistic link and switch costs are more complex – even for randomly generated instances. The benefits of redirecting traffic between switches, calculated in Section 3, are only positive if spare capacity can be found on existing links and switches to offset the additional cost of connection and switching.

Real link and switch costs increase in discrete steps. General principles are given in Chan (1998) and formulated in Gabrel *et al* (1999). Our cost functions are based on this approach.

7.2. Real-World Instances (RW)

Four real network problems were also studied. Actual network data in the form of node locations, traffic requirements and link/switch costs were provided as follows:

- Case 1: A Frame-Relay network of 78 nodes with estimated traffic flows, allowing the traffic matrix to be *approximated*.
- Case 2: A Frame-Relay network of 103 nodes with known (measured) traffic flows, allowing the traffic matrix to be *calculated*.
- Case 3: An ATM network of 221 nodes with unknown traffic flows. The traffic matrix is taken as being *constant* for all node pairs.
- Case 4: An IP network of 491 nodes with known (measured) traffic flows, allowing the traffic matrix to be *calculated*.

For reasons of commercial confidentially, it is not possible to release precise details of locations, traffic and costings. It is, however, acceptable to summarise results.

Three algorithms are considered and their complexity and accuracy compared. **RES**: Reduction to Exhaustive Search. **RDD**: Reduction to Double-Drop. **RSL**: Reduction to Switch Location.

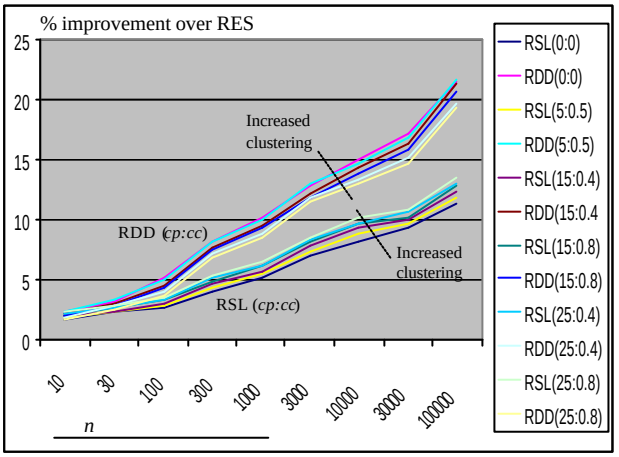

Figure 2. % Improvement of RSL & RDD over RES for different cluster values.

Based on each algorithm, coded in C++, running on a 2.8GHz Pentium IV processor, the optimisation limit values were derived (experimentally) as $n_{ES} = 12$ and $n_{DD} = 60$. These values permit run times of up to one day (*86,400* seconds). Figure 2 compares RSL, RES and RDD directly. There was some small variance for different network/traffic characteristics but the more significant trends are summarised in Figure 2 as percentage cost improvements of RSL and RDD over RES, averaged over all instances with clustering (*cp*, *cc*).

Essentially, RDD outperforms RSL, which in turn outperforms RES. RDD and RSL (by choosing appropriate values of n_{ES} and n_{DD}) can be constrained to approximately equivalent times. RSL is much faster – several orders of magnitude for the largest problems. RSL performed slightly better with increased levels of clustering and RDD slightly worse. All observed differences increase with larger values of *n*.

8. Conclusions

Theoretical, fixed-cost models of the network design process are simplistic. In the practical

design of a real network, both link and switch costs have to be considered and these costs are (at least partially) a function of (required) capacity. As this capacity depends upon the topology of the solution network, costs cannot be considered fixed and entered as input to a standard algorithmic solution. The further, implied difficulty that the cost function is not locally stable, and must be re-evaluated fully for each solution variant, increases the complexity considerably, particularly for large problems. Conventional construction or local search variants fail for one or both of these reasons.

Noting these objections, this paper initially considers two practical optimisation algorithms: exhaustive search (ES) and a doubly-iterative drop (DD). However, both have limits (n_{ES} and n_{DD}) on network size so additional techniques are needed to reduce larger problems to within their range. We consider three variants: reduce down to exhaustive search (RES), reduce down to double-drop (RDD) and reduce directly down to switch location (RSL). There is an additional heuristic simplification involved in RSL.

RES does not perform well, mainly due to a necessarily very small n_{ES}. RSL gives better results: its core network heuristic makes it the fastest of the three reductive approaches at the expense of some accuracy. RDD is the most accurate: $n_{DD} > n_{ES}$ outweighing the heuristic limitation of DD. If time permits, RDD would be the preferred method of solution for a large-scale network design problem. If less accurate results are required much faster (for example, if frequent re-optimisation is to be performed) then RSL is an acceptable compromise.

9. References

Aarts, E.H.L. and Lenstra, J.K., *Local Search in Combinatorial Optimization*, Wiley, 1997.

Chan, R., *Wide Area Network Design: Concepts & Tools for Optimization*, Morgan Kaufmann, 1998.

Dorigo, M., Di Caro, G. and Gambardella, L.M., "Ant Algorithms for Discrete Optimization," in *Artificial Life*, MIT Press, 1999.

Prim, R.C., "Shortest Connection Networks and some Generalizations," *Bell Systems Tech. J.* Vol. 36, pp1389-1401, 1957.

Gabrel, V., Knippel, A. and Minoux, M., "Exact solution of multicommodity network optimization problems with general step cost functions," *Operations Research Letters*, Vol. 25, pp15-23, 1999.

Garey, M.R. and Johnson, D.S., *Computers and Intractability: A guide to the theory of NP-completeness*, W.H. Freeman, New York, 1979.

Glover, F. and Laguna, M., "Tabu Search," in *Modern Heuristic Techniques for Combinatorial Problems*, Reeves, Ed. New York, Wiley, 1993, pp70-150.

Grout, V.(a), "Principles of Cost Minimisation in Wireless Networks", *J. Heuristics*, Vol. 11, Issue 2, March 2005.

Grout, V.(b), "Initial Results from a Study of Probability Curves for Shortest Arcs in Optimal ATSP Tours with Application to Heuristic Performance", *Proc. 20th British Combinatorial Conference* (*BCC 2005*), University of Durham, UK, 10th-15th July 2005.

Kershenbaum, A. and Chou, W., "A Unified Algorithm for Designing Multidrop Teleprocessing Networks", *IEEE Trans. Communications*, Vol. COM-22, No. 11, pp1762-1772, 1974.

Moon, J.W., "Counting Labelled Trees", *Canadian Mathematics Congress*, Montreal, 1970.

Winter, G., Periaux, J. and Galan, M. (Eds.), *Genetic Algorithms in Engineering and Computer Science*, Wiley, 1995.

Verkhovsky, B.S., and Polyakov, Y.S., "Algorithms for Optimal Switch Location: Concave Cost Functions," in *Advances in Decision Technology and Intelligent Information Systems, Vol. IV*, Lasker, Ed. Int. Inst for Advanced Studies in Systems Research and Cybernetics, pp16-20, 2003.